Data Structures

Instructor: Dr. Abdul Aziz.

# Class#1: Introduction

Objective of learning Data Structures: Managing Data in Memory.

Definition of Data Structures: Logical/Mathematical management of data.

**Visualization of arrays:**

Addresses:

* Logical
  + Indexing
  + Mapping
* Physical
  + Base Addresses + (sizeof(data type) \* index)
  + Longitude and Latitude

**Time-Space Trade off:**

The Time Complexity of a program is inversely proportional to its Space Complexity. An analogy of this is of car with various speeds. For attaining higher speed more fuel must be burned and vice versa.

**Types of Data Structures:**

* Linear
  + Array
  + Queue
  + Stacks
  + Link list (logically)
* Non-linear
  + Tree
  + Graph
  + Link list (physically)

To consider a data structure to be valid it must perform the following five operation:

1. Traversing/Accessing elements
2. Inserting
3. Editing
4. Deleting
5. Searching

Additionally, it can also include:

1. Merging
2. Sorting

**Coding Approach:**

Usually, the mistakes that is done many programmers is that they do not:

* Write basic procedures/ essence of the code is performing
* Documentation of projects

This results in difficulty in understanding the purpose and functionality of the code/program.

To avoid this, we can:

* Write pseudo code
* Document every functionality/feature/purpose/result of the program

# Class#2: Linked list

**Pre-class discussion:**

* Rule of Three

This refers to a safer approach to programming in C++. During defining of class, it is important to create:

* Destructor
  + To get rid of dangling pointers upon deletion of objects.
* Copy constructor
* Overloaded assignment operator

Other programming languages may be equipped with garbage collectors that prevent memory leakage. The reason why these functions must be made is that to avoid memory leakage and allow to deep copy and not shallow copy.

Problem analogy: Consider all copied objects as people living in a building having different apartments. If a mail is to be sent to one of people living in a building, then it would be unknown that to whom the mail is for as the address on the mail only consists of the location of the receiver and not the apartment number (considering that there is a building at that location). Moreover, if the building was to be destroyed, then all of the inhabitants of it would be deallocated. The similar case is made with the issue of shallow copy.

Shallow copy of objects causes them to be initialized on the same stack. If one of the objects is deleted/removed from the stack, then the other copies are also removed/deleted. The solution to this is deep copy the object by which the two objects are same/equal, but exist in the memory as a complete separate entity:

* For pointers having not pointing to the same heap space, but rather having separate memory allocation on heap.
* For being a separate entity on stack.
* Jagged Arrays

It is an array whose elements are also array, but with different/variable sizes. The issue regarding this form of data storage is that it may take up a big chunk of continues memory on the heap. It is also the case that the methods involving accessing the elements may be inefficient.

**Linked lists:**

Properties:

* It is a collection of data and pointers
* Physically Non-linear data structures
* Logically Linear data structures
* It can be a structure (in C/C++) or in class (in C++) that contains:
  + Data of any type.
  + Node pointer that contains the address of the next node (points to the next node)

Code and Visualization:

* Code
* class node{
* public:
* node():next(NULL){}
* int data;
* node\* next;
* };
* Diagram

A diagram of a data structure

Description automatically generated

Types of link lists:

* Singly linked
* Doubly linked
* Circularly linked
* Mixed

To fully define a structure/class for link list, construct, test and implement separate functions and then include them in the structure/class.

Functionality:

* Traversing / Displaying
* Editing
* Searching
  + Linear
  + Interpolation
* Insertion / Deletion / Merge
  + Cases
    - Start
    - Middle
    - End
* Sorting
  + Bubble
  + Selection
  + Insertion
  + Radix
  + Shell
  + Comb